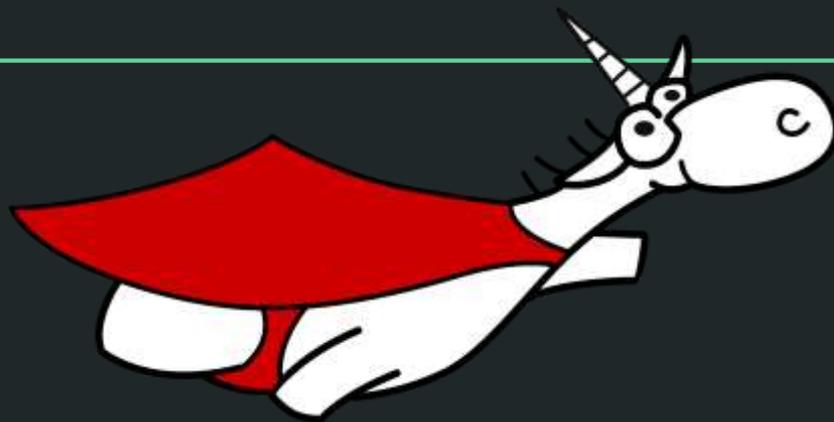


# Современный статический анализ кода: что умеет он, чего не умели линтеры

Павел Беликов  
PVS-Studio

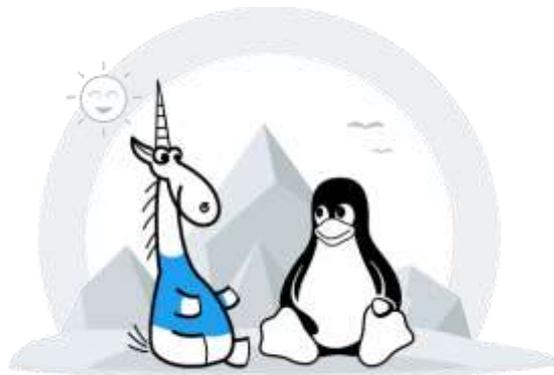
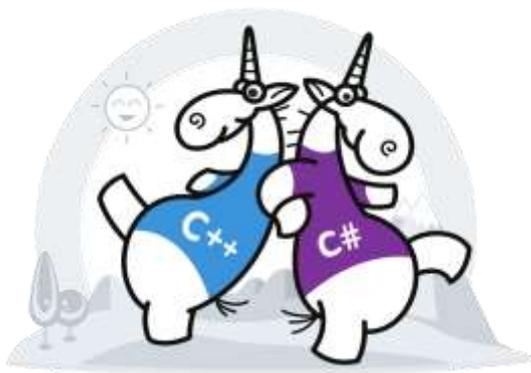


# Содержание

- чем должен и не должен заниматься статический анализ
- методики анализа: от первых утилит до современных анализаторов
- поиск по подстроке/регулярным выражениям
- pattern-based analysis
- symbolic execution
- межпроцедурный анализ
- современная инфраструктура анализаторов

# Немного о нас

- PVS-Studio - статический анализатор C, C++, C# кода
- Поддерживаются Windows и Linux (C, C++)
- Плагин для Visual Studio
- Средства для быстрой проверки (Standalone, pvs-studio-analyzer)



# Цели статического анализа

- поиск ошибок в логике программы
- поиск уязвимостей
- упрощение процесса разработки
- экономия

# Чем **не должен** заниматься статический анализ

- поиск “стилистических” ошибок
- соответствие каким-либо guideline
- предложениями переписать код на новый стандарт

Почему?

- для всего это есть отдельные инструменты
- в большом проекте таких срабатываний будут тысячи

# Как определить качество статического анализа

Существуют два типа ошибок, которым подвержены анализаторы:

- ошибки первого рода - false positives
- ошибки второго рода - false negatives

Ценность анализа в минимальном количестве ошибок первого рода.

Никто не будет пользоваться анализатором, который выдаёт 1000 ложных срабатываний на одну ошибку.

# С чего всё начиналось

- использование небезопасных функций
- возможные опечатки
- типовые ошибки, связанные с недостаточным знанием языка
- несоответствие принятому стилю кодирования
- такие ошибки ищутся и сейчас, отличаются методики поиска
- многие из них уже есть в компиляторах
- поиск по подстроке, регулярным выражениям или лексем ведёт к излишнему количеству ложных срабатываний

# Пример опечатки

## Ошибка из проекта **Linux**

```
char *stime[] = { "400ms", "5min", "10min", "15min",  
                 "20min", "25min", "30min" "No timeout" };
```

**Предупреждение PVS-Studio:** V653 A suspicious string consisting of two parts is used for array initialization. It is possible that a comma is missing. Consider inspecting this literal: «30min» «No timeout». lp8788-charger.c 657

# Поиск по регулярным выражениям

Некоторые их используют до сих пор. Пример из CppCheck:

```
<?xml version="1.0"?>
<rule version="1">
  <pattern> if \ ( ([!] )*?(strlen) \ ( \w+? \) ([>] [0] )*?\) { </pattern>
  <message>
    <id>StrlenEmptyString</id>
    <severity>performance</severity>
    <summary>Using strlen() to check if a string is empty is not
efficient.</summary>
  </message>
</rule>
```

Но у такого подхода серьезные проблемы с масштабируемостью.

# С чем же работает современный статический анализ

- представление кода в виде AST (или байткода, но это отдельный разговор)
- AST должно быть типизированным
- полная семантическая модель программы

# Pattern-based analysis

- *относительно* прост в реализации
- позволяет находить опечатки и последствия Copy-Paste
- в основе метода - поиск какого-то паттерна в АСТ
- для исключений может использоваться информация о типах и значениях



# Пример поиска паттерна

Ошибка из проекта **Oracle VM Virtual Box**

```
static const uint8_t g_acDaysInMonths[12];
static const uint8_t g_acDaysInMonthsLeap[12];

static PRTTIME rtTimeNormalizeInternal(PRTTIME pTime)
{
    ....
    unsigned cDaysInMonth = fLeapYear
        ? g_acDaysInMonthsLeap[pTime->u8Month - 1]
        : g_acDaysInMonthsLeap[pTime->u8Month - 1];
    ....
}
```

**Предупреждение PVS-Studio:** V583 The '?' operator, regardless of its conditional expression, always returns one and the same value: g\_acDaysInMonthsLeap[pTime->u8Month - 1]. time.cpp 453

# Пример паттерна с исключением

Ошибка из проекта **Serious Engine 1 v.1.10**

```
void CWorldEditorApp::OnConvertWorlds() {
    ....
    char achrLine[256];
    ....
    fsFileList.Open_t( fnFileList);
    while( !fsFileList.AtEOF()) {
        fsFileList.GetLine_t( achrLine, 256);
        // increase counter only for lines that are not blank
        if( achrLine != "") ctLines++;
    }
    fsFileList.Close();
    ....
}
```

**Предупреждение PVS-Studio:** V547 Expression 'achrLine != ""' is always true. To compare strings you should use strcmp() function. worldeditor.cpp 2254

# Пример паттерна с исключением

```
#define MY_STREQ(a,b) ((a) == (b) || strcmp(a, b) == 0)
```



# Паттерн с типами

## Ошибка из проекта **Wolfenstein 3D**

```
void BG_ParseConditionBits(  
    char **text_pp, animStringItem_t *stringTable,  
    int condIndex, int result[2] )  
{  
    ...  
    memset( result, 0, sizeof( result ) );  
    ...  
}
```

**Предупреждение PVS-Studio:** V511 The sizeof() operator returns size of the pointer, and not of the array, in 'sizeof (result)' expression.  
cgame bg\_animation.c 807

# Иногда и этого мало

## Ошибка из проекта **Linux**

```
#define CFS_FAIL_TIMEOUT(id, secs) \  
cfs_fail_timeout_set(id, 0, secs * 1000, CFS_FAIL_LOC_NOSET)  
  
#define OBD_FAIL_TIMEOUT(id, secs) \  
CFS_FAIL_TIMEOUT(id, secs)  
  
int ptl_send_rpc(struct ptlrpc_request *request, int noreply)  
{  
    . . . .  
    OBD_FAIL_TIMEOUT(OBD_FAIL_PTLRPC_DELAY_SEND,  
                    request->rq_timeout + 5);  
    . . . .  
}
```

**Предупреждение PVS-Studio:** V733 It is possible that macro expansion resulted in incorrect evaluation order. Check expression: request->rq\_timeout + 5 \* 1000. niobuf.c 637

# Symbolic execution

- самая сложная и самая полезная часть анализатора
- опирается на control flow analysis, data flow analysis
- позволяет осуществлять bounds checking, pointer analysis
- вычисление диапазонов переменной позволяет сэкономить на времени анализа

# Разыменование нулевого указателя

Ошибка из проекта **D programming language**

```
Expression *getVarExp(Loc loc, InterState *istate, Declaration *d, CtfeGoal
goal) {
    ....
    VarDeclaration *v = d->isVarDeclaration();
    if (v) {
        ....
    } else if (s) {
        if (s->dsym->toInitializer() == s->sym)
            ....
        else
            error(loc, "cannot interpret symbol %s at compile time",
                v->toChars());
    }
    ....
}
```

**Предупреждение PVS-Studio: V522**  
Dereferencing of the null pointer 'v' might  
take place. interpret.c 1711

# Разыменование нулевого указателя

- можно попытаться искать такие ошибки в AST, но не стоит
- вычислив значение указателя, мы можем находить более сложные ошибки

Пример:

```
void report_error() {  
    throw std::runtime_error("error");  
}
```

```
int* new_int() {  
    ....  
    int* p = some_condition ? new int : nullptr;  
    if (p == nullptr) // <= условие  
        report_error(); // <= не возвращает управления  
    *p = 0;           // <= p != nullptr, всё хорошо  
    return p;  
}
```

# Разыменование нулевого указателя

Ошибка из проекта **Clang**

```
Expected<std::unique_ptr<PDBFile>>
PDBFileBuilder::build(
    std::unique_ptr<msf::WritableStream> PdbFileBuffer)
{
    ....
    auto File = llvm::make_unique<PDBFile>(
        std::move(PdbFileBuffer), Allocator);

    File->ContainerLayout = *ExpectedLayout;

    if (Info) {
        auto ExpectedInfo = Info->build(*File, *PdbFileBuffer);
        ....
    }
}
```

**Предупреждение PVS-Studio: V522**  
Dereferencing of the null pointer  
'PdbFileBuffer' might take place.  
PDBFileBuilder.cpp 106

# Разыменование нулевого указателя

Ошибка из проекта **Unreal Engine 4**

```
void UGameplayStatics::DeactivateReverbEffect (....)
{
    if (GEngine || !GEngine->UseSound())
    {
        return;
    }
    UWorld* ThisWorld = GEngine->GetWorldFromContextObject (....);
    ....
}
```

**Предупреждение PVS-Studio:** V522 Dereferencing of the null pointer 'GEngine' might take place. Check the logical condition. gameplaystatics.cpp 988

# Control flow analysis

Какие конструкции не вернут управление:

- `return`
- исключение
- стандартная функция
- `[[noreturn]]`
- `__attribute__((noreturn))`
- `__declspec(noreturn)`
- создание временного объекта, деструктор которого бросает исключение
- вызов пользовательской функции, не возвращающей управление
- `goto`
- `break`, `continue`



Всё это нужно учитывать при вычислении возможных значений переменной

# Анализ функции

Ошибка из проекта **Chromium**

```
static int BlockSizeForFileType(FileType file_type) {
    switch (file_type) {
        ....
        default:
            return 0;
    }
}
```

```
static int RequiredBlocks(int size, FileType file_type) {
    int block_size = BlockSizeForFileType(file_type);
    return (size + block_size - 1) / block_size;
}
```

**Предупреждение PVS-Studio:** V609 Divide by zero. Denominator range [0..4096]. addr.h 159

# Bounds checking

Пример из мира 64-битных ошибок:

```
unsigned short f();
```

```
int g();
```

```
void test(void *p, void *p2) {
```

```
    int x = f();
```

```
    memset(p, 0, x); // <= всё нормально, x = [0; 65535]
```

```
    int y = g();
```

```
    memset(p2, 0, y); // <= подозрительно, int -> memsize
```

```
}
```

# Lifetime safety

Ошибка из проекта **Clang**

```
SingleLinkedListIterator<T> &operator++(int) {  
    SingleLinkedListIterator res = *this;  
    ++*this;  
    return res;  
}
```

**Предупреждение PVS-Studio:** V558 Function returns the reference to temporary local object: res. LiveInterval.h 679

# Путаница в new/malloc

Ошибка из проекта **Mozilla Thunderbird**

```
NPEror NPP_New(....)
{
    ....
    InstanceData* instanceData = new InstanceData;
    ....
    free(instanceData);
    ....
}
```

**Предупреждение PVS-Studio:** V611 The memory was allocated using 'new' operator but was released using the 'free' function. Consider inspecting operation logics behind the 'instanceData' variable. nptest.cpp 971

# Простой пример

```
void foo()  
{  
    int x = 1;  
    ....  
    if (x == 1) // <= всегда true, ошибка или нет?  
        return;  
}
```

- часто такой код используют для compile-time отключения функционала
- пример хорошо иллюстрирует неэффективность маленьких искусственных тестов

# Межпроцедурный анализ

## Рекурсивный анализ

- очень точный
- очень медленный



## Аннотирование

- можно составлять аннотации, анализируя тело функции
- для популярных библиотек составляются ручные аннотации
- отлично масштабируется



# Проблемы межпроцедурного анализа

- виртуальные методы
- указатели на функции
- межмодульный анализ
- динамическая линковка

# Анализ функции

- возможные значения
- область значений для параметров функции
- модификация аргументов, переданных по ссылке или указателю
- анализ на `noreturn`
- отсутствие побочных эффектов

# Аннотации: нет состояния

Ошибка из проекта **Clang**

```
std::pair<Function *, Function *>
llvm::createSanitizerCtorAndInitFunctions(
    ....
    ArrayRef<Type *> InitArgTypes, ArrayRef<Value *> InitArgs,
    ....)
{
    assert(!InitName.empty() && "Expected init function name");
    assert(InitArgTypes.size() == InitArgTypes.size() &&
        "Sanitizer's init function expects "
        "different number of arguments");
    ....
}
```

**Предупреждение PVS-Studio: V501**  
There are identical sub-expressions  
'InitArgTypes.size()' to the left and to the  
right of the '==' operator. ModuleUtils.cpp  
107

# Аннотации: строка в определённом регистре

Ошибка из проекта **CodeLite**

```
struct NodeJSHandle {  
    wxString type;  
    ....  
    bool IsString() const {return type.Lower() == "string";}   
    bool IsArray() const {return type.Lower() == "Array"; }  
};
```

**Предупреждение PVS-Studio:** V547 Expression 'type.Lower() == "Array"' is always false. NodeJSOutputParser.h 61

# Аннотации: значение должно быть использовано

Ошибка из проекта **Qt**

```
int main(int argc, char **argv)
{
    ....
    QByteArray arg(argv[a]);
    ....
    arg = arg.mid(1);
    arg.toLowerCase();
    if (arg == "o")
        ....
}
```

**Предупреждение PVS-Studio:** V530 The return value of function 'toLowerCase' is required to be utilized. main.cpp 72

# Аннотации: restrict аргументы

## Ошибка из проекта **Chromium**

```
void DiskCacheEntryTest::ExternalSyncIOBackground(...) {  
    ....  
    scoped_refptr<net::IOBuffer>  
        buffer1(new net::IOBuffer(kSize1));  
    scoped_refptr<net::IOBuffer>  
        buffer2(new net::IOBuffer(kSize2));  
    ....  
    EXPECT_EQ(0, memcmp(buffer2->data(), buffer2->data(), 10000));  
    ....  
}
```

# Аннотации: аргумент функции - количество байт

Ошибка из проекта **Miranda IM**

```
static BOOL ImageArray_Alloc(LP_IMAGE_ARRAY_DATA iad, int size)
{
    ...
    memset(&iad->nodes[iad->nodes_allocated_size],
           (size_grow - iad->nodes_allocated_size) *
sizeof(IMAGE_ARRAY_DATA_NODE),
           0);
    ...
}
```

**Предупреждение PVS-Studio:** V575 The 'memset' function processes '0' elements. Inspect the 'third' argument.  
clist\_modern/modern\_image\_array.cpp 59

# Пользовательские аннотации

- требуют большой работы по аннотированию всей кодовой базы
- индивидуальны для инструмента
- затрудняет чтение кода
- анализатор должен работать сразу



# Написание своих правил для анализатора

- хорошо для линтеров, проверяющих простые паттерны ошибок
- но плохо для статических анализаторов
- написать API, позволяющий работать со всеми видами анализа - огромная работа
- сделать его стабильным ещё сложнее
- пользователи не захотят писать полноценные правила для анализатора

# Ранжирование результатов анализа

- не все ошибки одинаково опасны
- некоторые диагностики могут тяготеть к ложным срабатываниям
- логично разделить ошибки на уровни
- отдельные категории диагностик: 64-битные, микрооптимизации

# Как осуществить такой анализ?

- анализ исходных файлов уже не работает
- отталкиваться приходится от конкретных компиляторов
- сборочные системы



# Как осуществить такой анализ?

- такое многообразие окружений накладывает свои требования
- необходимо уметь проверять проект, независимо от системы сборки
- необходимо парсить код для любого компилятора
- необходимо всё это тестировать (ура, больше исключений)

# Интеграция в процесс разработки

- статический анализатор - инструмент для разработчика
- инкрементальный анализ
- приемлемая скорость анализа
- подавление ложных срабатываний - assert, комментарии, база разметки

# Интеграция в QA

- запуск на сборочных серверах
- рассылка html-отчёта по почте
- интеграция в Code Quality платформы (например, SonarQube)

# Q&A

Почта: [support@viva64.com](mailto:support@viva64.com)

Блог на хабре: <https://habrahabr.ru/company/pvs-studio/>

Скачать PVS-Studio: <http://www.viva64.com/ru/pvs-studio/>



Для открытых некоммерческих проектов есть  
бесплатная лицензия